Supplementary Material for Decorrelated Batch Normalization

Lei Huang[†‡*] Dawei Yang[‡] Bo Lang[†] Jia Deng [‡]
[†]State Key Laboratory of Software Development Environment, Beihang University, P.R.China
[‡]University of Michigan, Ann Arbor

1. Derivation for Back-propagation

For illustration, we first provide the forward pass, then derive the backward pass. Regarding notation, we follow the matrix notation that all the vectors are column vectors, except that the gradient vectors are row vectors.

1.1. Forward Pass

Given mini-batch layer inputs $\{\mathbf{x}_i, i = 1, 2..., m\}$ where m is the number of examples, the ZCA-whitened output $\hat{\mathbf{x}}_i$ for the input $\mathbf{x}_i$ can be calculated as follows:

$$\mu = \frac{1}{m}\sum_{j=1}^{m}\mathbf{x}_j \tag{1}$$

$$\Sigma = \frac{1}{m}\sum_{j=1}^{m}(\mathbf{x}_j - \mu)(\mathbf{x}_j - \mu)^T \tag{2}$$

$$\Sigma = \mathbf{D}\Lambda\mathbf{D}^T \tag{3}$$

$$\mathbf{U} = \Lambda^{-1/2}\mathbf{D}^T \tag{4}$$

$$\tilde{\mathbf{x}}_i = \mathbf{U}(\mathbf{x}_i - \mu) \tag{5}$$

$$\hat{\mathbf{x}}_i = \mathbf{D}\tilde{\mathbf{x}}_i \tag{6}$$

where μ and Σ are the mean vector and the covariance matrix within the mini-batch data. Eqn. 3 is the eigen decomposition where $\mathbf{D}^T\mathbf{D} = I$ and Λ is a diagonal matrix where the diagonal elements are the eigenvalues. Note that $\tilde{\mathbf{x}}_i$ are auxiliary variables for clarity. Actually $\tilde{\mathbf{x}}_i$ are the output of PCA whitening. However, PCA whitening hardly works for deep networks as discussed in the paper.

*This work was mainly done while Lei Huang was a visiting student at the University of Michigan.

1.2. Back-propagation

Based on the chain rule and the result from [4], we can get the backward pass derivatives as follows:

$$
\begin{aligned}
\frac{\partial L}{\partial \tilde{\mathbf{x}}_i} &= \frac{\partial L}{\partial \hat{\mathbf{x}}_i}\mathbf{D} && (7)\\
\frac{\partial L}{\partial \mathbf{U}} &= \sum_{i=1}^{m} \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T (\mathbf{x}_i - \mu)^T && (8)\\
\frac{\partial L}{\partial \Lambda} &= (\frac{\partial L}{\partial \mathbf{U}})\mathbf{D}(-\frac{1}{2}\Lambda^{-3/2}) && (9)\\
\frac{\partial L}{\partial \mathbf{D}} &= \frac{\partial L}{\partial \mathbf{U}}^T \Lambda^{-1/2} + \sum_{i=1}^{m} \frac{\partial L}{\partial \hat{\mathbf{x}}_i}^T \tilde{\mathbf{x}}_i^T && (10)\\
\frac{\partial L}{\partial \Sigma} &= \mathbf{D}\{(\mathbf{K}^T \odot (\mathbf{D}^T \frac{\partial L}{\partial \mathbf{D}})) + (\frac{\partial L}{\partial \Lambda})_{diag}\}\mathbf{D}^T && (11)\\
\frac{\partial L}{\partial \mu} &= \sum_{i=1}^{m} \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}(-\mathbf{U}) + \sum_{i=1}^{m} \frac{-2(\mathbf{x}_i - \mu)^T}{m}(\frac{\partial L}{\partial \Sigma})_{sym} && (12)\\
\frac{\partial L}{\partial \mathbf{x}_i} &= \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}\mathbf{U} + \frac{2(\mathbf{x}_i - \mu)^T}{m}(\frac{\partial L}{\partial \Sigma})_{sym} + \frac{1}{m}\frac{\partial L}{\partial \mu} && (13)
\end{aligned}
$$

where L is the loss function, $\mathbf{K} \in \mathbb{R}^{d\times d}$ is 0-diagonal with $\mathbf{K}_{ij} = \frac{1}{\sigma_i - \sigma_j}[i \neq j]$, the $\odot$ operator is element-wise matrix multiplication, $(\frac{\partial L}{\partial \Lambda})_{diag}$ sets the off-diagonal elements of $\frac{\partial L}{\partial \Lambda}$ to zero, and $(\frac{\partial L}{\partial \Sigma})_{sym}$ means symmetrizing $\frac{\partial L}{\partial \Sigma}$ by $(\frac{\partial L}{\partial \Sigma})_{sym} = \frac{1}{2}(\frac{\partial L}{\partial \Sigma}^T + \frac{\partial L}{\partial \Sigma})$. Note that Eqn. 11 is from the results in [4]. Besides, a similar formulation to back-propagate the gradient through the whitening transformation has been derived in the context of learning an orthogonal weight matrix in [3].

1.3. Derivation for Simplified Formulation

For more efficient computation, we provide the simplified formulation as follows:

$$
\frac{\partial L}{\partial \mathbf{x}_i} = (\frac{\partial L}{\partial \tilde{\mathbf{x}}_i} - \mathbf{f} + \tilde{\mathbf{x}}_i^T\mathbf{S} - \tilde{\mathbf{x}}_i^T\mathbf{M})\Lambda^{-1/2}\mathbf{D}^T, \quad (14)
$$

where $\mathbf{f} = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}$, $\mathbf{S} = 2(\mathbf{K}^T \odot (\Lambda\mathbf{F}_c^T + \Lambda^{\frac{1}{2}}\mathbf{F}_c\Lambda^{\frac{1}{2}}))_{sym}$, $\mathbf{M} = (\mathbf{F}_c)_{diag}$ and $\mathbf{F}_c = \frac{1}{m}(\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T\tilde{\mathbf{x}}_i^T)$. The details of how to derive Eqn. 14 are as follows.

Based on Eqn. 4, 5, 8 and 9, we can get

$$
\begin{aligned}
\frac{\partial L}{\partial \Lambda} &= (\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T(\mathbf{x}_i - \mu)^T)\mathbf{U}(-\frac{1}{2}\Lambda^{-1})\\
&= -\frac{1}{2}(\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T\tilde{\mathbf{x}}_i^T)\Lambda^{-1}. \quad (15)
\end{aligned}
$$

Denoting $\mathbf{F}_c = \frac{1}{m}(\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T\tilde{\mathbf{x}}_i^T)$, we have

$$
\frac{\partial L}{\partial \Lambda} = -\frac{1}{2}m\mathbf{F}_c\Lambda^{-1}. \quad (16)
$$

Based on Eqn. 4 and 10, we have:

$$
\begin{aligned}
\mathbf{D}^T\frac{\partial L}{\partial \mathbf{D}} &= \Lambda^{\frac{1}{2}}\mathbf{U}^T(\frac{\partial L}{\partial U}^T\Lambda^{-\frac{1}{2}} + \sum_{i=1}^{m}\mathbf{D}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T\tilde{\mathbf{x}}_i^T)\\
&= m\Lambda^{\frac{1}{2}}\mathbf{F}_c^T\Lambda^{-\frac{1}{2}} + m\mathbf{F}_c. \quad (17)
\end{aligned}
$$

Based on Eqn. 11, we have :

$$
\begin{aligned}
\frac{\partial L}{\partial \Sigma} &= \frac{1}{2}\mathbf{D}\{(\mathbf{K}^T \odot (\mathbf{D}^T \frac{\partial L}{\partial \mathbf{D}})) + (\mathbf{K}\odot(\mathbf{D}^T\frac{\partial L}{\partial \mathbf{D}})^T)\}\mathbf{D}^T + \mathbf{D}(\frac{\partial L}{\partial \Lambda})_{diag}\mathbf{D}^T \\
&= \frac{1}{2}\mathbf{D}\{\mathbf{K}^T\odot m(\Lambda^{\frac12}\mathbf{F}_c^T\Lambda^{-\frac12}+\mathbf{F}_c)+\mathbf{K}\odot m(\Lambda^{-\frac12}\mathbf{F}_c\Lambda^{\frac12}+\mathbf{F}_c^T)\}\mathbf{D}^T + \mathbf{D}(\frac{\partial L}{\partial\Lambda})_{diag}\mathbf{D}^T \\
&= \frac{m}{2}\mathbf{D}(\mathbf{K}^T\odot(\Lambda^{\frac12}\mathbf{F}_c^T\Lambda^{-\frac12}+\mathbf{F}_c))\mathbf{D}^T + \frac{m}{2}\mathbf{D}(\mathbf{K}\odot(\Lambda^{-\frac12}\mathbf{F}_c\Lambda^{\frac12}+\mathbf{F}_c^T))\mathbf{D}^T + \mathbf{D}(\frac{\partial L}{\partial\Lambda})_{diag}\mathbf{D}^T \\
&= \frac{m}{2}\mathbf{U}^T(\mathbf{K}^T\odot(\Lambda\mathbf{F}_c^T+\Lambda^{\frac12}\mathbf{F}_c\Lambda^{\frac12}))\mathbf{U} + \frac{m}{2}\mathbf{U}^T(\mathbf{K}\odot(\mathbf{F}_c\Lambda+\Lambda^{\frac12}\mathbf{F}_c^T\Lambda^{\frac12}))\mathbf{U} + \mathbf{D}(\frac{\partial L}{\partial\Lambda})_{diag}\mathbf{D}^T \\
&= \frac{m}{2}\mathbf{U}^T\{\mathbf{K}^T\odot(\Lambda\mathbf{F}_c^T+\Lambda^{\frac12}\mathbf{F}_c\Lambda^{\frac12})+\mathbf{K}\odot(\mathbf{F}_c\Lambda+\Lambda^{\frac12}\mathbf{F}_c^T\Lambda^{\frac12}) - (\Lambda^{\frac12}\mathbf{F}_c\Lambda^{-\frac12})_{diag}\}\mathbf{U} \\
&= \frac{m}{2}\mathbf{U}^T\{2(\mathbf{K}^T\odot(\Lambda\mathbf{F}_c^T+\Lambda^{\frac12}\mathbf{F}_c\Lambda^{\frac12}))_{sym} - (\Lambda^{\frac12}\mathbf{F}_c\Lambda^{-\frac12})_{diag}\}\mathbf{U}. \\
&= \frac{m}{2}\mathbf{U}^T\{\mathbf{S}-\mathbf{M}\}\mathbf{U},
\end{aligned} \tag{18}
$$

where $\mathbf{S} = 2(\mathbf{K}^T\odot(\Lambda\mathbf{F}_c^T+\Lambda^{\frac12}\mathbf{F}_c\Lambda^{\frac12}))_{sym}$ and $\mathbf{M} = (\Lambda^{\frac12}\mathbf{F}_c\Lambda^{-\frac12})_{diag} = (\mathbf{F}_c)_{diag}$. Based on Eqn. 12, we have

$$
\begin{aligned}
\frac{\partial L}{\partial \mu} &= (\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i})(-\mathbf{U}) + (\sum_{i=1}^{m}\frac{-2(\mathbf{x}_i-\mu)^T}{m})(\frac{\partial L}{\partial \Sigma})_{sym} \\
&= -m\mathbf{f}\mathbf{U}
\end{aligned} \tag{19}
$$

where $\mathbf{f} = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}$. We thus have:

$$
\begin{aligned}
\frac{\partial L}{\partial \mathbf{x}_i} &= \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}\mathbf{U} + \frac{2(\mathbf{x}_i-\mu)^T}{m}(\frac{\partial L}{\partial \Sigma})_{sym} + \frac{1}{m}\frac{\partial L}{\partial \mu} \\
&= \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}\mathbf{U} + \tilde{\mathbf{x}}_i^T\mathbf{S}\mathbf{U} - \tilde{\mathbf{x}}_i^T\mathbf{M}\mathbf{U} - \mathbf{f}\mathbf{U} \\
&= (\frac{\partial L}{\partial \tilde{\mathbf{x}}_i} - \mathbf{f} + \tilde{\mathbf{x}}_i^T\mathbf{S} - \tilde{\mathbf{x}}_i^T\mathbf{M})\mathbf{U} \\
&= (\frac{\partial L}{\partial \tilde{\mathbf{x}}_i} - \mathbf{f} + \tilde{\mathbf{x}}_i^T\mathbf{S} - \tilde{\mathbf{x}}_i^T\mathbf{M})\Lambda^{-1/2}\mathbf{D}^T
\end{aligned} \tag{20}
$$

2. Computational Cost of DBN

In this part, we analyze the computational cost of DBN module for Convolutional Neural Networks (CNNs). Theoretically, a convolutional layer with a $d \times w \times h$ input, batch size of m, and d filters of size $F_h \times F_w$ costs $O(d^2 mhwF_hF_w)$. Adding DBN with a group size K incurs an overhead of $O(dKmhw + dK^2)$. The relative overhead is $\frac{K}{dF_hF_w} + \frac{K^2}{dmhwF_hF_w}$, which is negligible when K is small (*e.g.* 16).

Empirically, our unoptimized implementation of DBN costs 71ms (forward pass + backward pass, averaged over 10 runs) for *full* whitening, with a $64 \times 32 \times 32$ input, a batch size of 64. In comparison, the highly optimized 3×3 cudnn convolution [1] in Torch [2] with the same input costs 32ms.

Table A. Time costs (s/per epoch) on VGG-A and CIFAR datasets with different groups.

Methods	Training	Inference
BN	163.68	11.47
DBN-G8	707.47	15.50
DBN-G16	466.92	14.41
DBN-G64	297.25	13.70
DBN-G256	440.88	13.64
DBN-G512	1004	13.68

Table B. Time costs (s/per epoch) on residual networks and wide residual network on CIFAR.

	Training		Inference	
Method	BN	DBN-scale	BN	DBN-scale
Res-56	69.53	86.80	4.57	5.12
Res-44	55.03	72.06	3.65	4.36
Res-32	40.36	57.47	2.80	3.33
Res-20	25.97	42.87	1.94	2.44
WideRes-40	643.94	659.55	25.69	26.08
WideRes-28	440	457	36.56	38.10

Table A, B show the wall clock time for our CIFAR-10 experiments described in the paper. Note that DBN with small groups (*e.g*. G8) can cost more time than larger groups due to our unoptimized implementation: for example, we whiten each group sequentially instead of in parallel, because Torch does not yet provide an easy way to use linear algebra library of CUDA in parallel. Our current implementation of DBN has a low GPU utilization (*e.g*. 20%-40% on average), versus $95\%+$ for BN. Thus there is a lot of room for a more efficient implementation.

References

[1] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014. 3

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 3

[3] L. Huang, X. Liu, B. Lang, A. W. Yu, Y. Wang, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*, 2018. 2

[4] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. In *Proceedings of International Conference on Computer Vision, ICCV 2015*, 2015. 2